Licenciatura en ciencia de la computación

Algoritmo Euclideano

Matemática Computacional

Profesor:
Nicolas Thériault

Autor:
Sergio Salinas
Danilo Abellá

Introducción

1 Tiempos de ejecución de Algoritmo

A continuación se mostrarán los tiempos de ejecución apra cada valor de "n" y "p" respectivamente.

1.1 Tiempos de ejecución para valores de "n" y "p".

Costos computacionales de los algoritmos donde:

$$p \in \{1/10, 3/10, 1/2, 7/10, 9/10\}$$

para distintos valores de n:

n=18

p=0.1	57.0000000
p=0.3	43.0000000
p=0.5	41.0000000
p=0.7	128.0000000
p=0.9	40.0000000

n=660

p=0.1	18815.0000000
p=0.3	18698.0000000
p=0.5	26974.0000000
p=0.7	18506.0000000
p=0.9	21156.0000000

n=8000

p=0.1	2890310.0000000
p=0.3	3851808.0000000
p=0.5	5042122.0000000
p=0.7	6085456.0000000
p=0.9	6920520.0000000

1.2 Costo computacional para n = 1000.

Costos computacionales de los algoritmos para n = 1000 para distintos valores de p. Tiempos de espera para los siguientes valores de "p":

p=0 34094.0000000
p=0.1 31610.0000000
p=0.2 37190.0000000
p=0.3 55043.0000000
p=0.4 47476.0000000
p=0.5 45258.0000000
p=0.6 46934.0000000
p=0.7 56155.0000000
p=0.8 56874.0000000
p=0.9 47229.0000000
p=1 48239.0000000

2 Formulación experimentos

Se experimentó con los tiempos de ejecución del algoritmo
para distintos valores de "n" y "p", pudiendo ver así que tan influyentes son los datos de las respectivas variables, y tener una mejor idea de la eficiencia del programa.
Para ello se dió uso de gráficos para poder estudiarlo mejor.

3 Repretesentación de grafo utilizada

Se utilizo una matriz de adyacencia para representar el grafo debido a que en C se puede manejar sencillamente como un arreglo de dos dimensiones. De esta forma escribir y recorrer la matriz fue tan sencillo como recorrer una matriz de largo n, donde n es la cantidad de vértices que coincide con la cantidad de filas y columnas de va a tener la matriz.

4 Algoritmo de busqueda de camino

La principal razón porque la la búsqueda de ciclo de euler da un ciclo un de euler es debido a que en el grafo generado todas los vértices tienen grado par.

Para crear el algoritmo se utilizo una función recursiva para recorrer el grafo y borrar las aristas entre los vértice y una estrategia de backtracking para volver un paso atrás en el caso de que haya llegado a un vértice sin aristas y aun hayan vértice que no se han recorrido. El algoritmo hace lo siguiente.

Al inicio de la función recursiva se posiciona en la fila y columna 0, a partir de ahí comienza a recorrer la fila hasta encontrar un 1, eso significa que hay una arista entre el vértice que esta representando la fila y el vértice que representa la columna, por lo cambia el 1 por un 0, al ser una matriz simétrica se borra en la posición transpuesta también. Después se va a la intercambia la posición de la fila por la columna y se empieza a recorrer esa fila desde la posición 0. el algoritmo continua así hasta que se recorre toda la matriz solo tenga 0 o hasta que se llegue a una fila llena de unos.

En el caso de que llegue a una fila llena de ceros significa que o que se encontró el camino euleriano o que se borraron todas las aristas de ese vértice y aun faltan aristas por recorrer en los otros vértices.

En el primer caso se termina el algoritmo y se muestra el camino. En el segundo se debe hacer un backtracking, para ello se ve el arreglo en el que se guardaba el camino euleriano y se vuelven a poner 1 donde se habían por 0, luego se va a última fila que se recorrió, pero en vez se partir desde la posición 0, esta vez se parte desde la posición siguiente al 0 que se borro, de esta forma se puede seguir recorriendo el grafo sin problemas hasta que ya se haya recorrido todas las aristas.

5 Información de Hardware y Software

5.1 Notebook - Danilo Abellá

5.1.1 Software

- SO: Xubuntu 16.04.1 LTS
- GMP Library
- Mousepad 0.4.0

5.1.2 Hardware

- AMD Turion(tm) X2 Dual-Core Mobile RM-72 2.10GHz
- Memoria (RAM): 4,00 GB(3,75 GB utilizable)
- Adaptador de pantalla: ATI Raedon HD 3200 Graphics

5.2 Notebook - Sergio Salinas

5.2.1 Software

- SO: ubuntu Gnome 16.04 LTS
- Compilador: gcc version 5.4.0 20160609
- Editor de text: Atom

5.2.2 Hardware

- Procesador: Intel Core i7-6500U CPU 2.50GHz x 4
- Video: Intel HD Graphics 520 (Skylake GT2)

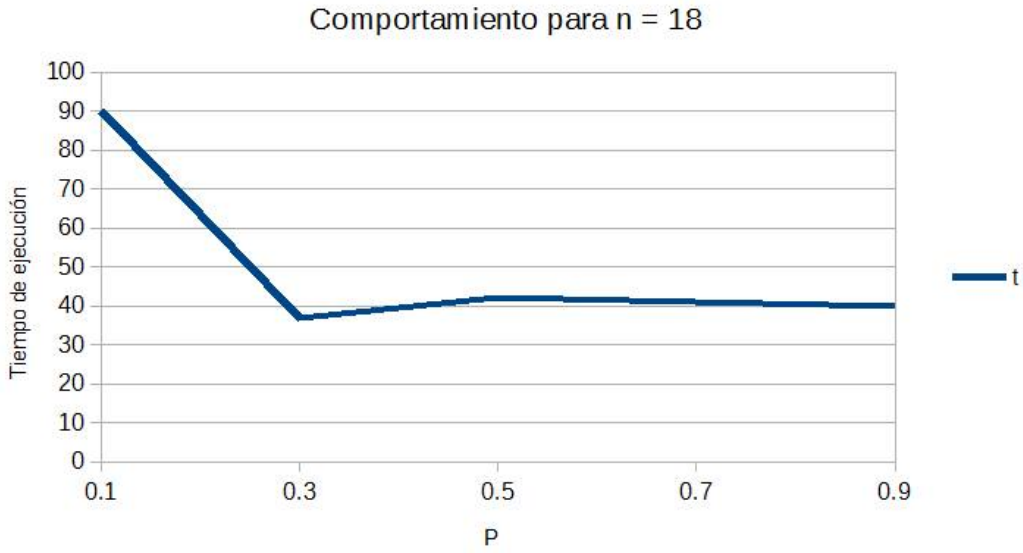
6 Curvas de desempeño de resultados

6.1 Comportamiento para los valores de "p" pedidos.

Teniendo un valor de "n" vastante bajo la gráfica sólo mostró un cambio brusco del p=0.1 al p=0.3.

Tabla :

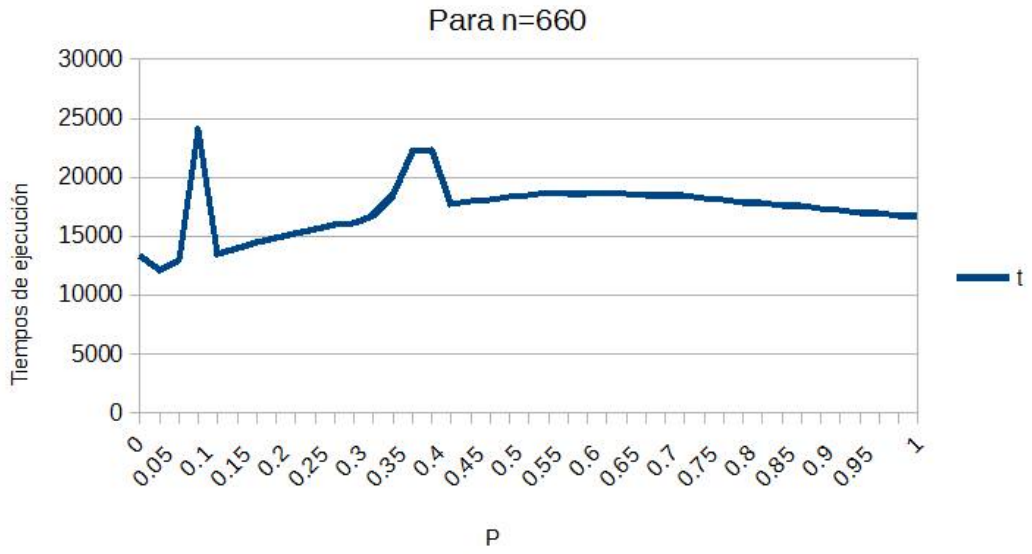
p	t
0.1	35216
0.3	39415
0.5	54627
0.7	57627
0.9	47100



Con "n" un valor considerablemente mayor la gráfica no mostró resutlados iniciales tan bruscos como en el caso anterior.

Tabla :

p	t
0	13303
0,025	12085
0.05	12971
0,075	24086
0.1	13481
0.0125	14026
0.15	14426
0.0175	14847
0.2	15222
0.0225	15570
0.25	15951
0.0275	16142
0.3	16713
0.0325	18646
0.35	22169
0.0375	22335
0.4	17718
0.0425	17997
0.45	18101
0.0475	18305
0.5	18476
0.0525	18687
0.55	18552
0.0575	18622
0.6	18679
0.0625	18599
0.65	18467
0.0675	18426
0.7	18408
0.0725	18244
0.75	18051
0.0775	17792
0.8	17787
0.0825	17612
0.85	17537
0.0875	17375
0.9	17175
0.0925	17007
0.95	16957
0.0975	16761
1	16721



Tomando "n" como un número muy grande la grafica se mostró ascendente.

Tabla :

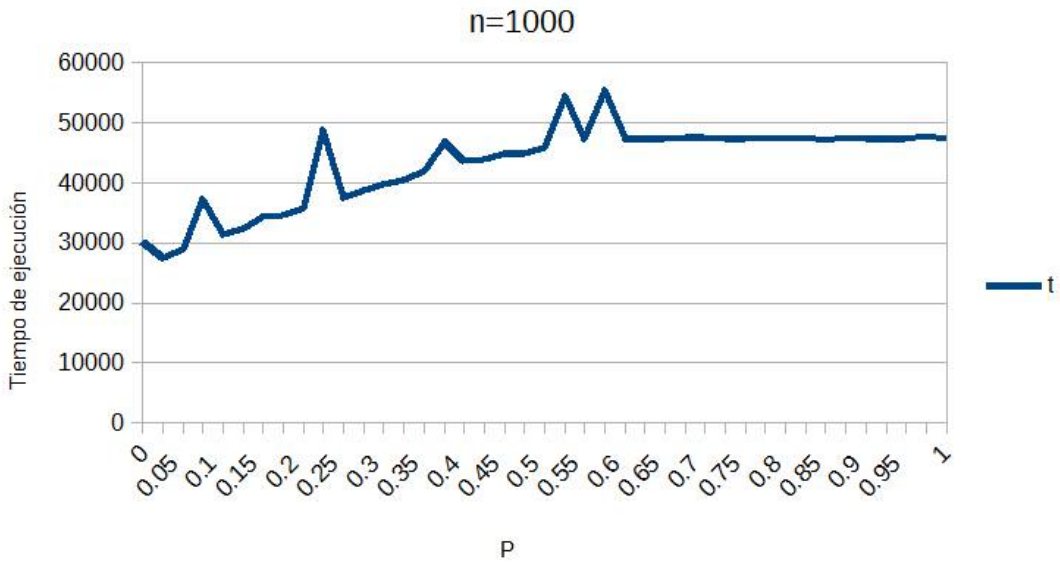
p	t
0	2018934
0,025	2018063
0.05	1953712
0,075	2310997
0.1	2520134
0.0125	2536516
0.15	2882736
0.0175	3006946
0.2	2962661
0.0225	3125603
0.25	3550458
0.0275	3594267
0.3	3844001
0.0325	3993128
0.35	4326592
0.0375	4233552
0.4	4316045
0.0425	4423400
0.45	4624221
0.0475	4695363
0.5	4884622
0.0525	5085860
0.55	5118988
0.0575	5458820
0.6	5507959
0.0625	5497049
0.65	5657734
0.0675	5888063
0.7	5914492
0.0725	6735436
0.75	8285270
0.0775	6327969
0.8	6439842
0.0825	6631450
0.85	6781128
0.0875	6795587
0.9	6980377
0.0925	7214039
0.95	7331707
0.0975	7255347
1	7482635

6.2 Costos computacionales de los algoritmos para n = 1000 para distintos valores de "p".

Para el caso pedido de n = 1000 se obtuvo una grafica ascendente hasta que p = 0.625.

Tabla :

p	t
0.1	90
0.3	37
0.5	42
0.7	41
0.9	40



7 Conclusiones

Con los datos recopilados de los gráficos podemos concluir que mientras mayor sea el valor de ”n” mayor tiempo de ejecución se requiuere.

También cave decir que independiente de ”n” mientras mayor sea el valor de ”p” el tiempo de ejecución también tiende siempre a aumentar , lo que quiere decir que ambas variables influyen en el tiempo de ejecución.

Otro punto a considerar es que cuando ”n” no es tan grande (apróx. menor a 1000) la variable ”p” no es tan influyente , sin embargo cuando ”n” toma un valor grande (apróx. mayor a 1000) ”p” tiene mucha mas influencia en el tiempo de ejecución.